

Towards a Complete Framework for Virtual Data Center Embedding

M. P. Giles

National Institute of Technology Calicut

Kozhikode, Kerala, India

Email: giles_p140059cs@nitc.ac.in

Abstract

Cloud computing is widely adopted by corporate as well as retail customers to reduce the upfront cost of establishing computing infrastructure. However, switching to the cloud based services poses a multitude of questions, both for customers and for data center owners. In this work, we propose an algorithm for optimal placement of multiple virtual data centers on a physical data center. Our algorithm has two modes of operation - an online mode and a batch mode. Coordinated batch and online embedding algorithms are used to maximize resource usage while fulfilling the QoS demands. Experimental evaluation of our algorithms show that acceptance rate is high - implying higher profit to infrastructure provider. Additionally, we try to keep a check on the number of VM migrations, which can increase operational cost and thus lead to service level agreement violations.

1. Introduction

In the last few years, cloud computing has become the backend of all the service oriented IT businesses. The conventional service providers (SP) are able to significantly reduce their capital expenditure (CapEx) by switching to virtualized IT infrastructure. The time to establish (ToE) infrastructures of any desired size has come down from several months to a few minutes/hours of automated provisioning of virtual machines (VMs) in physical infrastructure providers' premises and installing the application stack on these virtual infrastructures [8] using tools like Puppet, Ansible, and Vagrant.

With the new model of service provisioning, the business role of conventional SPs has got divided among infrastructure providers (InP), who expose a software abstraction of the underlying hardware or software resources; and service providers, who use these abstractions to build their own application stack

to offer services to the end users. Virtualization of the hardware is the prime technology that enabled the cloud computing paradigm. Though the concept of virtualization is more than two decades old, cloud computing has leveraged the exploitation of the potential of virtualization at various levels viz. hardware, development platform and software. As of now, almost every domain of computing infrastructure, including computing, storage, network, operating systems and applications, have been virtualized to maximize the resource utilization and reduce the ToE. The InPs maintain a pool of resources to be abstracted and multiple instances of user machines/ applications run on these resources, typically called *slices*.

Network virtualization, though late entrant in cloud computing, has key role in complying with the SLA between InP and SPs. The bandwidth requirement for internal traffic is growing fast against that for external traffic in cloud data centers [10]. From the business perspective, virtualizing the network is crucial because the cost of networking is escalating against the cost of other equipments and the ratio of network to compute is going up, especially in big data applications. Literature show that, NV is killer application for the software defined networks (SDN) technology. In this paper, we address the problems associated with sharing compute and network resources of an InP, in particular, that of a datacenter infrastructure. In multi-tenant data centers, multiple service deploy their virtual infrastructure (VI) for service provisioning. The service level agreement (SLA) defines the QoS expectations of the service providers and the penalty for the violation of the same. Hence, it is critical for the InP to provide isolation between these VIs, in terms of performance, disruption, and security.

In this paper we address the virtual data center embedding (VDCE) problem in cloud data center environment [7]. In VDCE, embedding of virtual networks (VNs) and virtual machines (VM) are considered simultaneously. A virtual datacenter (VDC), like any

other DC, has VMs connected through virtual switches using virtual links of specific bandwidth and delay constraints. VDCE refers to the class of algorithms which optimally superimpose multiple VDCs on a DC, satisfying the SLA. [14], [17]. The problem is important because, many customers (i.e. SPs) try to create or migrate their topology intact (including switches) to a cloud for reasons like failure recovery, scaling or economic saving.

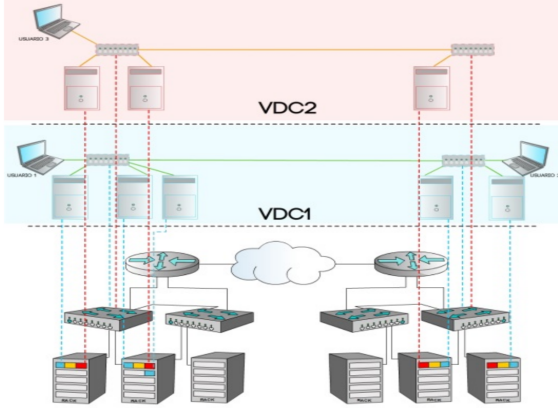


Figure 1. Virtual Data Center Embedding [7]

Formally, the problem of VDCE, is defined as follows. Let $G^s = (N^s, E^s, C^s, \phi_a^s, \delta_b^s, \xi_c^s)$ be a physical DC and there is a set of VDC requests $\mathcal{R} = \{G_u^v = (N_u^v, E_u^v, C_u^v, \phi_{a_u}^v, \delta_{b_u}^v, \xi_{c_u}^v) : u = 1, 2, \dots, m\}$. Table 1 gives the description of the variables used. The problem is to find an optimal placement of VDCs on DC while satisfying the SLAs with the tenants $u = 1, 2, \dots, m$. In other words, find a mapping from VDC set to DC

$$\mathcal{F}_u^C : C_u^v \rightarrow C^{s'}$$

$$\mathcal{F}_u^N : N_u^v \rightarrow N^{s'}$$

$$\mathcal{F}_u^E : E_u^v \rightarrow \mathcal{P}^{s'}$$

for $u = 1, 2, \dots, m$ where $\mathcal{P}^{s'} \subset (E^s)^k, 1 \leq k \leq |N^s|, N^{s'} \subseteq N^s, C^{s'} \subseteq C^s$, under the following conditions

- 1) Virtual machines (VM) are mapped to servers/physical machines (PM).

$$\mathcal{F}_u^C(c_1) = c \in C^{s'} \quad (1)$$

- 2) Not more than one node of a virtual request is mapped to a physical (substrate) switch (pSwitch).

$$\mathcal{F}_u^N(n_1) = \mathcal{F}_u^N(n_2) \implies n_1 = n_2 \quad (2)$$

and hence, at most one virtual link (vLink) of a request is mapped to a path. Moreover, virtual

Var	Description
\mathcal{R}	Set of serviceable VDC requests
N^s	Set of substrate switches
E^s	Set of substrate links
C^s	Set of servers
\mathcal{P}^s	Set of all paths in substrate network
N_u^v	Set of virtual switch of request u
E_u^v	Set of virtual links of request u
C_u^v	Set of VMs of the request u
$\phi_a^s / \delta_{a_u}^v$	Value of PM / VM attribute a
$\delta_b^s / \delta_{b_u}^v$	Value of pSwitch / vSwitch attribute b
$\xi_c^s / \xi_{c_u}^v$	Value of pLink / vLink attribute c
q^C	Set of machine attributes
q^N / q^E	Set of switch/machine/link attributes
\mathcal{P}_{kl}	Set of paths in SN between k and l
$y_{ij,kl}^u$	1 if vLink ij of $u \in \mathcal{R}$ is mapped to n th path kl .
x_{ik}^u	1 if vSwitch i of $u \in \mathcal{R}$ is mapped to a pSwitch k , Else 0
w_{ik}^u	1 if VM i of $u \in \mathcal{R}$ is mapped to a PM k , Else 0
Z_u	1 if a request u is mapped. Else 0.
p_{kln}^e	1 if e is in the n th path of kl . Else 0

Table 1. Notations used

edge switches are mapped only to edge switches of the physical topology.

- 3) Sum of capacity demands of VMs does not exceed the total capacity of the PM to which they are mapped

$$\sum_{u \in \mathcal{R}: \mathcal{F}_u^C(k)=s} \phi_{a_u}^v(k) \leq \phi_a^s(s) \quad \forall s \in C^s, \forall a \in q^C \quad (3)$$

- 4) Similarly, sum of capacity demands of virtual switches (vSwitch) should be within that of the pSwitch to which they are mapped

$$\sum_{u \in \mathcal{R}: \mathcal{F}_u^N(k)=n} \delta_{b_u}^v(k) \leq \delta_b^s(n) \quad \forall n \in N^s, \forall b \in q^N \quad (4)$$

- 5) The sum of attribute demands of vLinks do not exceed the total capacity of the physical / substrate link (pLink) to which they are mapped.

$$\sum_{u \in \mathcal{R}: \mathcal{F}_u^E(l)=p; e \in p; p \in \mathcal{P}^s} \xi_{c_u}^v(l) \leq \xi_c^s(e) \quad \forall e \in E^s, \forall c \in q^E \quad (5)$$

Figure 1 illustrates how two VDC requests are embedded on a physical data center.

The node mapping problem - mapping virtual nodes to physical nodes - was proven to be NP-Hard [4] [6], [18]. The problem is reducible from the famous Multi-way Separator Problem [4] which is NP-Hard. Optimal assignment of links with functional constraints, in a graph, where the nodes are already assigned, is still NP-hard [6], [18] and is similar to Unsplittable Flow

Problem (UFP) [11] / Multi-Commodity Flow (MCF) problems. Many heuristic solutions were proposed for the problem.

We propose a VDCE algorithm that employs suitable technique for embedding requests depending on the availability of DC resources. The model employs mathematical programming for embedding a group of requests, in one shot. Local search heuristic is used for online embedding of requests to reduce the average waiting time of the requests. The online embedding reduces the number of VM migrations by not consolidating the VMs, as proposed by many solutions found in literature.

Rest of this paper is organized as follows. Section 2 gives a brief review of the relevant work in this area. Section 3 explains the proposed algorithm in detail. In Section 4 we discuss experimental setup for evaluation of our algorithm and the results of these experiments. Section 5 concludes the paper.

2. Related Work

Virtual Network Embedding problem, for efficient placement of virtual networks on physical network, is being studied for several years. Algorithm to efficiently map virtual nodes to the physical nodes, to satisfy various objectives had been studied. Single shot embedding of both nodes and edges were also studied [5] [9]. In practice, exact algorithms for the problem apply for small networks only. A comprehensive survey of classic virtual network embedding techniques is given by Belbekkouche et al. [5] and Fischer et al. [9]. Fischer et al. [9] proposed a taxonomy for VNE and classified them as centralized vs distributed, static vs dynamic and redundant vs concise. The centralized, dynamic and redundant algorithms are particularly of interest because they apply to the present day data center networks, especially the software defined networks

The VDCE problem is different from VNE because the latter cares about the efficient placement of VMs with multiple constraints. Efficient placement of VMs on servers is an independent problem by itself. To the best of our knowledge, there are only a few proposals for VDC resource allocation in cloud computing centers. Sivaranjini et al. [16] and Correa et al. [7] give surveys on the existing virtual data center embedding algorithms. Papagianni et al. [12] proposed a VNE for cloud computing environment viz. Networked Cloud Mapping (NCM), that handle both compute and network node assignments. The mixed integer programming (MIP) model attaches multiple property vectors for the nodes including the type of nodes, capacity etc.

Zhani et al. [20] proposed a VDC planner to embed virtual data center network and machines on a substrate data center. The primary consideration of the proposal is migration-awareness. There are three scenarios considered in the proposal - an initial deployment, handling up/down scaling of VDC and a dynamic consolidation (reoptimization) for power saving. The experimentations are carried out on the VL2 topology [10]. Three stage virtual data center embedding algorithm by Rabbani et al. [13] maps virtual machines, switches and links in sequential stages. The heuristic algorithm reduces the server fragmentation, communication cost and the resource utilization.

Amokrane et al. [3] give a resource allocation technique for virtual data center spanning over distributed substrate data centers. The proposed method has two stages - a VDC partitioning and a partition embedding. In partitioning, the VDC requests are partitioned to minimize the inter-data center bandwidth. In the second phase, the partitions are mapped to data centers satisfying capacity constraints. The integer linear program (ILP) formulation of both phases ensures exact solution for reduced bandwidth. Venice is a reliability aware VDC embedding algorithm proposed by Zhang et al. [19]. The authors proved that computing the availability of VDC is a hard problem. The reliability-aware embedding is handled using a heuristics and a consolidation to handle the frequent entry and exit of requests. Wang et al. [17] proposed a heuristic framework for the VDCNE problem - *Presto*. The framework uses Blocking Island (BI) paradigm for improving the accuracy and speed of embedding. However, the link and node embeddings are uncoordinated and the heuristic based solutions are sub-optimal.

Our survey shows that algorithms are either exact algorithms that suffer from sluggishness or heuristic algorithms which leave the DCs under-provisioned. A suboptimal embedding might be possible using heuristics. But, such an embedding may lead to rejection of other peer requests, resulting poor QoE (Quality of Experience) of customers. Moreover, most of the heuristics fail if the residual resources are spread over multiple fragments. Hazzles of live VM migration dismiss the possibility of frequent remapping of the incumbent nodes. There had been solutions which use VM migrations to find optimality, neglecting the cost of migration. So, we believe that, a well coordinated hybrid technique can find an optimal solution in less time. We propose a mixed approach with suitable modes from mathematical programming and local search heuristics to maximize the resource usage with limited VM migrations.

3. Proposed Algorithm

The proposed algorithm works in two different modes - online mode or batch mode - based on the rate of fragmentation of resources within data center. Different techniques are used to perform batch embedding and online embedding. If sufficient resources are available, individual arriving requests are embedded immediately with local search method to avoid reoptimization of the entire mapping. Online embedding is done also when one or more embedded requests exit and if the total residual resources are above a threshold (t^1) defined in terms of the smallest pending request. Online embedding is attempted in case of failure of hardware components also. Frequent reoptimization of the entire embedding or consolidation may cause many VM migrations. It may cause unprecedented delay in the communication within a virtual network, if a tenant *vSwitch* is mapped to a farther *pSwitch* after reoptimization. Batch embedding of multiple requests is performed by a mixed integer programming model. Batch embedding is performed if there are sufficient resources to embed multiple VDC requests or the total fragmented resources are above a threshold vector t^2 defined in terms of the resource requirement of the largest pending request. Threshold vector has values t_c^2, t_n^2, t_e^2 the residual values of servers, switches and links.

Initially, the residual resource vectors t_c, t_n, t_e are initialized with the capacity vectors of the data center. If there are multiple pending requests and the total demand is less than t_c, t_n, t_e we try a batch embedding. If the sum of resource demand vectors of all requests exceed t_n, t_e for nodes and links respectively, and multiple requests can be embedded, then a batch embedding is attempted on a selected subset request. The subset has the high priority jobs. Details of the online embedding and batch embedding algorithms are given below.

3.1. Online Embedding

Online embedding is the most common occurrence in our model. In online mode, an embedding is attempted whenever arrival of a new request or an exit of embedded request happens. Specifically, an online embedding is attempted when:

- (i) a new request arrives and the local remapping can enable the embedding of the request; or
- (ii) there is failure of some servers, nodes and links and VDC components already mapped to the failed part of the substrate requires a remapping; or

- (iii) a mapped virtual data center requires scaling up and free resources are not available in the neighbourhood.

online embedding attempts to find a local solution with minimum modification to the existing mappings. If there are multiple waiting requests, we prioritize the requests based on the time to expiry, size of the VDC and expected time of incumbency. If a fragment (connected component) of the DC graph has sufficient resources to embed the selected request then, an embedding is attempted by simple heuristics. We use a local search and swap technique to do online embedding. A temporary mapping of the new request is made in the following order - VMs, Switches, Links. The temporary mapping is allowed to have capacity violations. Then swapping is attempted between already mapped VMs/Switches in locations where the temporary mapping has least violations. If a solution is derived by few number of swappings (limited by the number of swaps required) the temporary mapping is made permanent. If neither of the above are possible the algorithm tries to embed the next request.

3.2. Batch Embedding

Batch embedding is attempted when a set of requests are available for the first time or when there are enough fragmented resources for embedding a new/differed request for which it is hard to find a local solution. In our algorithm we focus on maximizing the acceptance rate of requests which positively affects the revenue of InP and availability of the cloud service. The batch embedding algorithm works as follows.

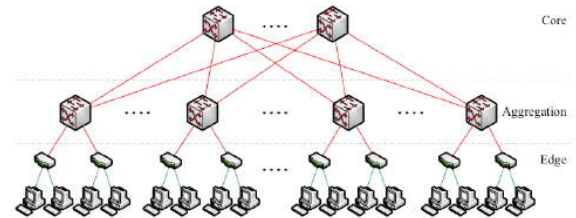


Figure 2. Data Center Topology

We formulated the batch embedding problem as a mixed integer program (MIP). The model restricts at most one *VS*witch of a request on an *PS*witch. However, this restriction does not apply to embedding of VMs on PMs. Multiple VMs can be embedded in a single PM. An edge switch of a VDC request is embedded on an edge switch of the substrate topology (Figure 2). Any *VLink* is embedded on a single path between the nodes onto which the end virtual devices

are mapped. The MIP model for batch embedding is as follows

Equation (6) tries to maximize the number VDCs embedded, while minimizing the number of VM migrations, subject to the constraints (7) - (16)

$$\text{Maximize } \sum_{u \in \mathcal{R}} Z_u - r_1 - \frac{r_2}{f} \quad (6)$$

where r_1 and r_2 are the normalized migration distances of VMs and vSwitches already embedded. Parameter f decides the weight of penalty for vSwitch migration relative to VM migration.

Constraint (7) and (8) ensures that when a VDC u is embedded all its vSwitches and VMs are also embedded.

$$\sum_{k \in N^s} x_{ik}^u = Z_u \quad \forall i \in N_u^v, u \in \mathcal{R} \quad (7)$$

$$\sum_{k \in C^s} w_{ik}^u = Z_u \quad \forall i \in C_u^v, u \in \mathcal{R} \quad (8)$$

As per the definition, at most one vSwitch of virtual request is embedded on a pSwitch. Constraint (9) affirms this rule.

$$\sum_{i \in N_u^v} x_{ik}^u \leq 1 \quad \forall k \in N^s, u \in \mathcal{R} \quad (9)$$

To ensure that a virtual link is embedded only once and not split across multiple path, constraint (10) is used. There can be exponentially many paths between any two nodes of a graph. We consider paths of length utmost 4 and other paths are not included in \mathcal{P}_{kl} .

$$\sum_{kl \in \{N^s \cup C^s\} \times \{N^s \cup C^s\}} \sum_{n \in \mathcal{P}_{kl}} y_{ij,kl,n}^u = Z_u \quad \forall ij \in E_u^v, u \in \mathcal{R} \quad (10)$$

The linking constraint (11) ensures that, whenever a pair of vSwitches are mapped, the vLink between them are mapped to at most one path between the selected pSwitches.

$$\sum_{n \in \mathcal{P}_{kl}} y_{ij,kl,n}^u - x_{ik}^u x_{jl}^u = 0 \quad \forall ij \in E_u^v, u \in \mathcal{R} \quad (11)$$

For any server in the DC, there is one (Figure 2) or a constant number of switches in the neighbourhood. The topology in use has only one neighbour. Hence the constraint for embedding a (vSwitch, VM) link is given as

$$\sum_{kl \in E^s \cap (N^s \times C^s)} y_{ij,kl}^u - x_{ik}^u w_{jl}^u = 0 \quad \forall ij \in E_u^v \cap (N_u^v \times C_u^v), u \in \mathcal{R} \quad (12)$$

The constraint (13) assures that the sum of resource demands of VMs mapped to a PM does not exceed its corresponding resource capacity.

$$\sum_{u \in \mathcal{R}} \sum_{i \in C_u^v} w_{ik}^u \phi_a(i) \leq \phi_a(k) \quad \forall k \in C^s, \forall a \in |q^C| \quad (13)$$

Similarly, constraint (14) ensures that the sum of resource demands of vSwitches mapped to a pSwitch does not exceed its corresponding resource capacity.

$$\sum_{u \in \mathcal{R}} \sum_{i \in N_u^v} x_{ik}^u \delta_b(i) \leq \delta_b(k) \quad \forall k \in N^s, \forall b \in |q^N| \quad (14)$$

Similar to the above the sum of link resource demands of all vLinks, mapped to a pLink, should be less than the capacity of the substrate.

$$\sum_{u \in \mathcal{R}} \sum_{ij \in E_u^v} y_{ij,kl,n}^u p_{kl,n}^e \xi_c(ij) \leq \xi_c(e), \quad \forall e \in E^s, \forall c \in |q^E| \quad (15)$$

Following constraint put bounds on values that the variables can assume.

$$Z_u \in \{0, 1\}, y_{ij,kl,n}^u \in \{0, 1\}, x_{ik}^u \in \{0, 1\}, w_{ik}^u \in \{0, 1\} \quad (16)$$

Apart from the constraints given above, VDC requests may have specific demands on other QoS parameters like latency, location, distance between VMs etc. The following set of constraints apply to those requests which are in need of such special consideration. If the VDC requests $u \in \mathcal{R}' \subset \mathcal{R}$ want to limit the maximum latency between adjacent nodes (switch/VM) to d_u , the following constraint (17) is added in respect of them.

$$\left(\sum_{vw \in P_{kl,n}} d_{vw} \right) y_{ij,kl,n}^u \leq d_u \quad \forall ij \in E_u^v, \forall u \in \mathcal{R}' \subset \mathcal{R} \quad (17)$$

where d_{vw} is the delay of a physical link (v, w)

A vital requirement of VDC placement in cloud environment is the freedom to specify the location for embedding a VM. Customarily, the need arises from the locality of other nodes, ease of access etc. This is a major requirement in virtual clusters performing big data processing, to improve computation time. The following constraint is used to specify the exact substrate node k on which a virtual node i should be embedded.

$$w_{ik}^u = Z_u \quad (18)$$

A more flexible method would be specify a possible subset of PMs to embed a given VM. Optional constraint (19) limits the embedding VM i on a PM from

a set of servers $C^{s'} \subset C^s$

$$\sum_{k \in C^{s'}} w_{ik}^u = Z_u \quad (19)$$

As and when needed, batch embedding is done for compaction of the unusable fragmented resources. Such a re-embedding of incoming and mapped VDC request is VM migration aware and hence tries to minimize/limit the cost of changes in the existing mapping. We assume that virtual machines of different sizes have varying cost of migration. Hence costly migrations are avoided to the extent possible. If remapping a *VSwitch* is necessary, the new *PSwitch*, to which the mapping is done, is not far from the already mapped *PSwitch*. This feature keeps a check on the number of VM migration and the extra internal traffic.

Variable	Values
PM CPU Cores	8
PM Memory	16384
pSwitch memory	100
pLink bandwidth (core - aggregation)	10000
pLink bandwidth (edge - aggregation)	1000
pLink bandwidth (edge - server)	1000
No of VMs	40 - 100
VM cores	1 - 2
VM Memory	256 - 512
No of vSwitches	5 - 20
vSwitch memory	10 - 25
vLink bandwidth	5 - 200
Duration of incumbancy	10 - 90

Table 2. Simulation parameters

4. Preliminary Results

We developed the simulation environment in python, with FNSS toolchain [15] and NetworkX library in the backend, for managing the substrate and virtual networks. The batch embedding is implemented with a commercial solver, CPLEX [1]. Our algorithm interfaces with CPLEX using python concert API of CPLEX. Online embedding heuristic is implemented with python, using NetworkX library.

We use the fat tree topology [2] shown in (Figure 2) to represent the substrate data center. For simplicity, we consider bandwidth of links and memory capacity of switches as the attributes for the network. Number of CPU cores and Memory are considered as the attributes of Servers and VMs. VDC requests have random sized tree/fat tree topology - a most common case in big data processing clusters. The topologies were generated using the topology generator of FNSS toolchain [15]. Simulation parameters are randomly distributed with ranges as given in Table 2. The

arrivals of VDCN requests are determined by a Poisson distribution ranging from 1 to 10 requests per 100 time units.

The acceptance rate with respect to varying rate of arrival is shown in Figure 3. The figure clearly exposes the advantage of our strategies over MIP or heuristics solution used singly. Poor acceptance rate of heuristic can be accounted to the sub optimal solutions of heuristic methods. This leaves a lot of resources in data center unused. MIP solutions are applied in batches and hence the acceptance largely rely on the instantaneous availability of resources whereas, the resources remain unused between successive embedding sessions.

Number of VM migrations is a measure of quality of the VDCE solutions. Figure 4 shows that our algorithm reduces the rate of VM migrations. Higher number of migrations in pure MIP solutions is intuitive from the nature of the algorithm. Every time a new batch is embedded, the existing VMs get migrated with highest probability, to reach optimality. With pure MIP model it is hard to achieve optimal solution without VM migration in datacenter with fragmented resources. The remapping in heuristic algorithms are due to the periodic consolidation of resources, which is inevitable.

5. Conclusions

Our algorithm is suitable for optimally embedding virtual data centers on a physical data center. The VDCE problem is important in placement of VMs with or without specific topology demands. The proposed multi-mode algorithm maximizes the resource utilization over time by the online embedding, without waiting for embedding in batches. In case fragmentation is high, an MIP based batch re-embedding is applied to alleviate the same. Sluggish ILP-based exact algorithm for batch embedding is the major limitation of our proposal. We are looking for faster mathematical programming techniques or near-optimal heuristics that can overcome the limitation.

References

- [1] Ilog cplex optimization studio v12.6.3, 2016.
- [2] Mohammad Al-Fares, Alexander Loukissas, and Amin Vahdat. A scalable, commodity data center network architecture. *SIGCOMM Comput. Commun. Rev.*, 38(4):63–74, August 2008.
- [3] A. Amokrane, M.F. Zhani, R. Langar, R. Boutaba, and G. Pujolle. Greenhead: Virtual data center embedding across distributed infrastructures. *IEEE Transactions on Cloud Computing*, 1(1):36–49, Jan 2013.

Algorithm 1 Virtual Data Center Embedding

```

1: procedure VDC_EMBED( $G^s, \mathcal{R}$ )
2:    $ts_c$  = residual server resource vector
3:    $ts_n$  = residual switch resource vector
4:    $ts_e$  = residual link resource vector
5:   while  $|\mathcal{R}| \neq 0$  do
6:     Initialize the set  $\mathcal{R}'$ 
7:     Calculate the thresholds  $(t_c^2, t_n^2, t_e^2)$  and
       $(t_c^1, t_n^1, t_e^1)$ 
8:     if  $(ts_c, ts_n, ts_e) \geq (t_c^2, t_n^2, t_e^2)$  then
9:       call batch_embedding( $G^s, \mathcal{R}'$ )
10:    else if  $((t_c^1, t_n^1, t_e^1) \leq (ts_c, ts_n, ts_e) <$ 
       $(t_c^2, t_n^2, t_e^2))$  then
11:      Priority sort that set  $\mathcal{R}$ .
12:      Select  $G_u^v \in \mathcal{R}$  s.t.  $t_{exp}^u \leq t_{curr} + TD$ 
13:      call online_embedding( $G^s, G_u^v$ )
14:    end if
15:  end while
16: end procedure

17: procedure BATCH_EMBEDDING( $G^s, \mathcal{R}$ )
18:   Formulate the MIP (Section 3.2)
19:   Solve the MIP (with a solver)
20:   for VDC  $u$  not embedded do
21:     Add  $u$  back to  $\mathcal{R}$ 
22:   end for
23: end procedure

24: procedure ONLINE_EMBEDDING( $G^s, G_u^v$ )
25:   if residual_resource( $C$ )  $\not\geq$  requirement( $G_u^v$ ) then
26:     Create temporary VDC placement  $C$ 
27:     Confirm the embedding if possible with local
      search.
28:   else
29:     Select servers with least capacity violation.
30:     Find the  $vi$  the amount of violations.
31:     Select smallest set of VMs with capacity  $\geq vi$ 
32:     Swap with VM group where the difference
      in capacity is atleast  $vi$ 
33:   end if
34: end procedure

```

- [4] David G Anderson. Theoretical approaches to node assignment. *Unpublished Manuscript*, 2002.
- [5] Abdelouab Belbekkouche, Md Mahmud Hasan, and Ahmed Karmouch. Resource discovery and allocation in network virtualization. *IEEE Communications Surveys and Tutorials*, 14(4):1114–1128, 2012.
- [6] Mosharaf Chowdhury, Fady Samuel, and Raouf Boutaba. Polyvine: Policy-based virtual network embedding across multiple domains. In *Proceedings of the Second ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures, VISA '10*, pages 49–56, New York, NY, USA, 2010. ACM.
- [7] E.S. Correa, L.A. Fletscher, and J.F. Botero. Virtual data center embedding: A survey. *Latin America Transactions, IEEE (Revista IEEE America Latina)*, 13(5):1661–1670, May 2015.

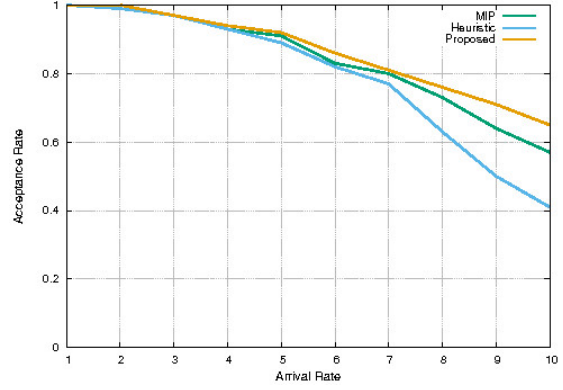


Figure 3. Acceptance rate for varying arrival rates

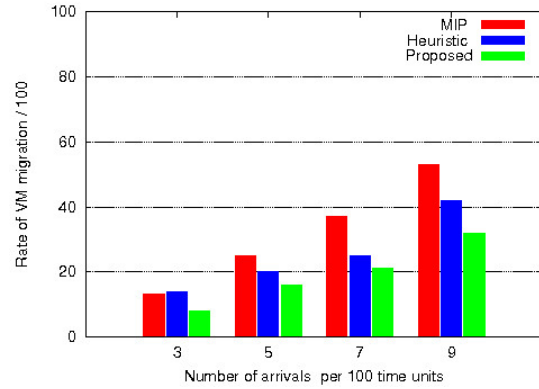


Figure 4. % of VM migrations for different arrival rates

- [8] Y. Demchenko, F. Turkmen, C. de Laat, C. Blanchet, and C. Loomis. Cloud based big data infrastructure: Architectural components and automated provisioning. In *2016 International Conference on High Performance Computing Simulation (HPCS)*, pages 628–636, July 2016.
- [9] A. Fischer, J.F. Botero, M. Till Beck, H. de Meer, and X. Hesselbach. Virtual network embedding: A survey. *Communications Surveys Tutorials, IEEE*, 15(4):1888–1906, Fourth 2013.
- [10] Albert Greenberg, James R. Hamilton, Navendu Jain, Srikanth Kandula, Changhoon Kim, Parantap Lahiri, David A. Maltz, Parveen Patel, and Sudipta Sengupta. V12: A scalable and flexible data center network. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication, SIGCOMM '09*, pages 51–62, New York, NY, USA, 2009. ACM.
- [11] A. Haider, R. Potter, and A. Nakao. Challenges in resource allocation in network virtualization. In *Proc. of 20th ITC Specialist Seminar on Network Virtualization*, Hoi An, Vietnam, 2009.
- [12] C. Papagianni, A. Leivadeas, S. Papavassiliou,

- V. Maglaris, C. Cervello-Pastor, and A. Monje. On the optimal allocation of virtual resources in cloud computing networks. *IEEE Transactions on Computers*, 62(6):1060–1071, June 2013.
- [13] M.G. Rabbani, R. Pereira Esteves, M. Podlesny, G. Simon, L. Zambenedetti Granville, and R. Boutaba. On tackling virtual data center embedding problem. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 177–184, Ghent, Belgium, May 2013.
- [14] R. V. Rosa, C. E. Rothenberg, and E. Madeira. Virtual data center networks embedding through software defined networking. In *2014 IEEE Network Operations and Management Symposium (NOMS)*, pages 1–5, Poland, May 2014.
- [15] Lorenzo Saino, Cosmin Cocora, and George Pavlou. A toolchain for simplifying network simulation setup. In *Proceedings of the 6th International ICST Conference on Simulation Tools and Techniques, SIMUTOOLS '13*, Cannes, France, 2013. ICST.
- [16] B. Sivaranjani and P. Vijayakumar. A technical survey on various vdc request embedding techniques in virtual data center. In *2015 National Conference on Parallel Computing Technologies (PARCOMPTECH)*, pages 1–6, Bangalore, India, Feb 2015.
- [17] T. Wang, B. Qin, and M. Hamdi. An efficient framework for online virtual network embedding in virtualized cloud data centers. In *2015 IEEE 4th International Conference on Cloud Networking (CloudNet)*, pages 159–164, Canada, Oct 2015.
- [18] Minlan Yu, Yung Yi, Jennifer Rexford, and Mung Chiang. Rethinking virtual network embedding: Substrate support for path splitting and migration. *SIGCOMM Comput. Commun. Rev.*, 38(2):17–29, March 2008.
- [19] Qi Zhang, M.F. Zhani, M. Jabri, and R. Boutaba. Venice: Reliable virtual data center embedding in clouds. In *2014 Proceedings IEEE INFOCOM*, pages 289–297, Toronto, April 2014.
- [20] M.F. Zhani, Qi Zhang, G. Simon, and R. Boutaba. Vdc planner: Dynamic migration-aware virtual data center embedding for clouds. In *2013 IFIP/IEEE International Symposium on Integrated Network Management (IM 2013)*, pages 18–25, Ghent, Belgium, May 2013.